Direct Manipulation of Isosurfaces and Cutting Planes in Virtual Environments

Tom Meyer
Department of Computer Science
Box 1910
Brown University
Providence RI 02912
(401) 863-7693
twm@cs.brown.edu

Al Globus
Computer Science Corporation
MS T045-1
NASA Ames Research Center
Moffett Field CA 94035-1000
(415) 604-4404
globus@nas.nasa.gov

ABSTRACT

This paper presents some novel interface techniques for the visualization of scalar field data in virtual reality (VR) environments. In particular, we discuss the direct manipulation of isosurfaces and cutting planes and give general guidelines for design of user interfaces and algorithms for scientific visualization in virtual environments. This work was motivated by the need to visualize scalar field data from computational fluid dynamics (CFD) calculations interactively in the Virtual Wind Tunnel at NASA Ames, but the techniques are applicable to other problem areas, including medical imaging and geological and mathematical visualization.

KEYWORDS: scientific visualization, virtual reality, direct-manipulation interfaces, scalar fields, isosurfaces, cutting planes

INTRODUCTION

Data sets in scientific visualization often take the form of 2D and 3D scalar fields. For example, height data on a 2D topographic map, temperatures in the fluid flow over a wing, and the density information in a magnetic resonance imaging (MRI) brain scan are all scalar fields.

Scalar field data consists of a real-valued function *f* defined over a 2D or 3D coordinate space. 3D scalar fields, which are especially difficult to understand both because they are not easily rendered on a 2D screen and because they often contain enormous amounts of data, are the focus of this paper. We describe two new user interface techniques for the interactive visualization of cutting planes and isosurfaces. Such interactive interfaces will become increasingly important for visualizations in such areas as geology, chemistry, engineering, physics, biology, and mathematics.

We discuss some general issues of user-interface design in virtual reality, and their application to computational fluid dynamics visualization. In doing so, we describe the specific problems involved in manipulating visualizations of large 3D data sets, especially in immersive virtual environments. Techniques for dealing with these problems include gestural manipulation techniques and the use of algorithms which take the available time for computation into account (real-time algorithms).

PRIOR WORK

Many visualization systems, such as AVS [13] and FAST [1], support the specification and viewing of cutting planes and isosurfaces in a desktop environment. While these may be sufficiently interactive for desktop work, they do not support the real-time interaction speeds required for VR. Additionally, they do not provide direct manipulation interfaces to these visualization techniques, using sliders and text entry instead for tool manipulation.

The Virtual Wind Tunnel project [5], under the direction of Steve Bryson at NASA Ames Research Center, provides a number of direct manipulation interfaces for interactive viewing of vector field data in a virtual environment. It supports visualizations of steady and unsteady flows using streamlines, streaklines, and pathlines. Another tool designed for the exploration of scientific data in a virtual environment is the University of North Carolina's Nanomanipulator [12], which lets the user visualize the scalar field of the height of an atomic surface in a virtual environment, incorporating both visual displays and force feedback.

The Brown University Graphics Group has done considerable work on the specification and design of 3D widgets used for direct-manipulation in modeling and animation [7]; however, little work has been done there on designing 3D widgets specifically for scientific visualization. In this paper, we examine relevant design criteria and apply these criteria to create two useful widgets for visualizing scalar fields in a virtual environment.

VIRTUAL REALITY ENVIRONMENTS

We define a VR environment as a computer-generated simulation that is immersive and creates an actual sense of presence for the user, typically achieved through head-tracking stereo displays and gestural input techniques. In order to maintain this sense of presence, the simulation must react to the user's movements without noticeable lag and maintain a frame rate of at least 8-10 frames per second [4].

The Virtual Wind Tunnel system at NASA consists of a head-mounted display and glove input device. The system included several tools for visualizing vector fields. Because scientists using the system requested the addition of tools for the visualization of scalar fields, we began working on highly-interactive interfaces to these visualization techniques.

SCALAR FIELDS

Scalar fields are often represented by points sampled at discrete intervals in a 3D grid. Each subvolume bounded by these sample points is known as a *voxel* (volume element), which is analogous to a 2D pixel in a bitmap. Grids may be stretched and curved to place more sample points in areas of interest, such as areas in which the field values change rapidly. The actual types of mappings between grid (computational) space and physical space are beyond the scope of this paper, but it is important to note that the data need not be sampled uniformly.

Because 3D scalar fields are not directly viewable – they consist solely of numbers in a 3D grid – we need to transform them into useful visualizations. The three techniques commonly used to visualize 3D scalar field data are:

- Cutting Planes

  A cutting plane samples the 3D scalar field along a slice through the data and then renders the data by coloring it or generating contour lines in the slice. With such a slice or cutting plane, the user can view a manageable subset of the data and can explore the field by sweeping the plane through the space.

- Isosurfaces

  An isosurface, the 3D equivalent of a single contour line, is the continuous surface that passes through all voxels containing the specified value. An isosurface is commonly generated as a polygonal surface, using an algorithm such as Marching Cubes [8]. Isosurfaces provide a more global view of a particular value in the field than cutting planes, but do not let the user see how quickly the data changes, as in the cutting plane technique, and also can take some time (from a few seconds to a few minutes to compute). Isosurfaces renderings are often used on MRI data to generate pictures of individual organs.

- Volume Rendering

  In a volume rendering, the data set is traversed in a manner similar to X-ray imaging. As a ray is projected through the data set, it accumulates color and opacity from the scalar field, yielding areas that are more or less transparent. Because of the difficulty of comprehending the resulting picture, however, it is often necessary to render a number of images from different angles, and interactively rotate through them. Since this technique is very time-consuming and cannot presently be done at rates anywhere near the ten frames per second required for VR, volume rendering is not discussed further.

The use of cutting planes and isosurfaces to visualize scalar fields is well suited to virtual-reality environments. Because the data is inherently 3D, viewing it in 3D makes sense. The head-tracked display lets the user walk around and through the data set, and 3D hand tracking lets the user probe the data gesturally using direct-manipulation interfaces. However, the speed and ease with which the data can be explored accentuates any delays in computing or rendering the visualizations.

These visualization techniques become cumbersome when applied to the large data sets common to most scientific problems. A scalar field consisting of a cube with 100 voxels on a side contains a million points. Large data sets not only consume memory but require significant CPU time to compute isosurfaces and cutting planes. Even when a visualization can be computed rapidly, it may contain too many polygons to be rendered at interactive frame rates. Performance is worse when data changes over time, such as in an unsteady fluid flow or MRI data from a beating heart. In the following sections of this paper, we address ways to improve the speed of these visualization techniques to make them usable in virtual environments.

DESIGN PROCESS

What Do Scientists Want?

In the interface design process, it is necessary to determine what the users (in this case scientists) actually need. Springmeyer, et al., conduct an empirical study of the work process of scientists, and give design recommendations for scientific data analysis tools [11].

- *Facilitate active exploration*. Tools should allow for interactive qualitative and quantitative exploration of data.
- *Capture the context of analysis*. Tools should support the annotation of data.

- *Decrease cumbersome maneuvering*. Tools should minimize navigation requirements, and provide support for the culling of large data sets.

The Virtual Wind Tunnel project is intended to be used for the rapid interactive exploration of data, and as such primarily addresses the first and third recommendations. The second, the annotation of visualization data, is beyond the scope of this paper.

Widgets

Specification of 3D cutting planes and isosurfaces can be difficult with traditional 2D user interfaces consisting of textual input and 2D sliders, since these are difficult to correlate with the actual visualization parameters.

Interactive 3D *widgets*, defined as a combination of geometry and behavior to control application objects, provide powerful direct-manipulation techniques for viewing and manipulating data. Ideally, we would like a set of easy-to-use widgets that allow for the direct manipulation of cutting planes and isosurfaces and facilitate the rapid exploration of the data.

The design process for these widgets falls into two basic parts:

- *Understanding the design space.* This involves examining the implicit design rules used in the Virtual Wind Tunnel and applying accepted user-interface design principles to this environment.
- *Creating and refining the widgets*. This involves working in a rapid-prototyping environment, where a widget can be quickly created and modified in response to user feedback, thus shortening the design cycle (the amount of time between making a change and seeing its effects). This can be achieved through graphical interfaces, interpreted languages, dynamically-loaded libraries, or just by working on a small subset of the system at any given time.

General Widget Design Guidelines

A useful set of general criteria for widget design is [10]:

- *Self-disclosure.* The geometry of the widget should indicate its behavior through handles and other cues (affordances).
- *Implicit versus explicit control of parameters.* Widgets usually explicitly control parameters, but may implicitly control additional parameters in order to reduce the complexity of the interface.

- *Degrees of freedom.* Removing unnecessary degrees of freedom and simplifying the interaction yield a more effective widget.
- *Intended use.* A widget used for artistic purposes has different requirements from one intended for engineering and may need to incorporate different interaction techniques.

These guidelines are effective during the high-level user-interface design.

Design Principles for the Virtual Wind Tunnel

Through informal discussions and the exploration of the existing Virtual Wind Tunnel environment, we identified the following stylistic guidelines for our widgets.

- *Limited range of gestures.* Interfaces with many possible different gestures tend to be difficult to learn. With more gesture types, gesture recognition becomes more difficult and error-prone; in addition, people's hand coordination varies greatly, and there are only a few gestures that all users can make easily. Therefore, the Virtual Wind Tunnel incorporates only three gestures: fist, open hand, and point, representing grabbing, letting go, and menu selection, respectively.
- *Highlight when grabbable*. When the user is within grabbing range of an object, the object highlights, indicating that it may be grabbed.
- *Do not attach objects to the hand*. It should always be possible to let go of a widget and stop interacting with it.
- *Do not hide the data*. In a scientific visualization environment, widgets that obscure the data should be avoided. Balancing this requirement against the need for self-disclosing widgets can, however, be difficult.

These stylistic guidelines, in combination with the design principles given previously, provide a means to evaluate potential widgets, before they are designed, built, and tested for useability.

ISOSURFACES

The most common isosurface manipulation interface is a simple 2D slider controlling the isosurface threshold. An isosurface computation algorithm such as Marching Cubes [9] [2] then performs a global search through the data to compute all the isosurfaces. This computation can take anywhere from a few seconds to several minutes, depending on the hardware and size of the data set. The delay can distract users and impede comparisons during isosurface threshold sweeps.

Time-Critical Algorithms

To address rendering speed for isosurfaces, we investigated time-critical algorithms: algorithms guaranteed to finish within a specified time, that begin with a simplified or degraded answer, which which is then refined incrementally as time permits. For example, an algorithm might choose to compute shorter streamlines while the user is manipulating them, in order to maintain the current interaction rates.

The Isosurface Algorithm

A good algorithm for direct manipulation might allow the user to probe the data set interactively and would draw the isosurface which passes through the probe, updating at highly interactive rates. In order to ensure that the visualization can be examined rapidly, we can consider ways in which the visualization could be degraded so as to limit computation and rendering time and still yield sufficient context for the data.

A simple algorithm to examine areas of likely importance is to search in the immediate vicinity of the probe, extend the search as far as time permits, and then draw only the locally generated surface. Such an algorithm, known as the “tracking” algorithm, was developed by Bloomenthal [3] for modeling implicit surfaces. This algorithm generates a set of voxels containing the surface by starting at the voxel containing the sample point. It then determines which of the adjacent voxels the surface continues through, and adds those to the list if they have not already been examined. One can then use a standard voxel-polygonization technique to generate triangles approximating the surface in that voxel. This technique has the attractive properties that it always returns a surface through the sample point, it is fast, and always generates the surface nearby before generating far away points.

Because the algorithm only performs a local search, it is very fast and can display significant portions of the isosurface at highly interactive rates. However, since it does not perform a global search, it finds only the local closed isosurface that passes through the sample point: other disconnected surfaces at the same threshold value are not drawn. Because we are trying to help scientists with their initial understanding of their scalar field, and because it is always possible to perform a global search to generate the complete surface at a greater cost of time, this drawback seems acceptable.

The Isosurface Widget

We considered several ways of using the widget. The most obvious one exploits the many degrees of freedom of a position tracker and lets the user interactively position a probe around which the isosurface is generated. This technique is intended for an initial exploration of a data set, so as to better understand its overall structure. While the probe is

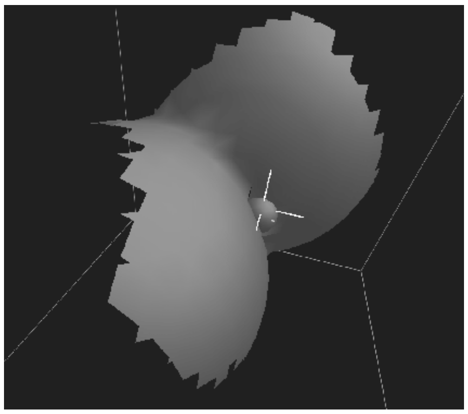

Figure 1: The isosurface while it is being moved

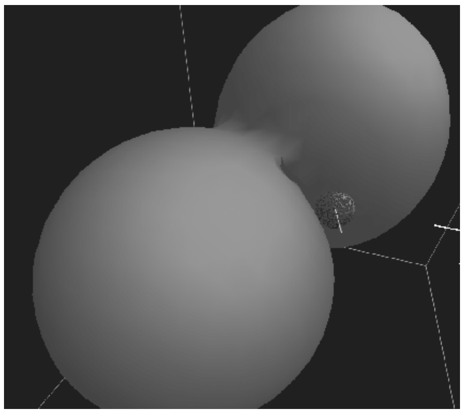

Figure 2: The completed isosurface

being moved, the value it samples is constantly changing, so the surface never has additional time to complete the computation. This means that the surface remains ragged while it is being moved and then expands to close out when the user releases the probe, as shown in Figures 1 and 2. The scalar value of the generated isosurface is also displayed, offset from the widget slightly so that it is more easily seen.

Since this type of interaction felt very natural, we tried to think of other ways in which the user would like to interact with the isosurface. Since users customarily interact with isosurfaces using a slider interface, we looked at ways to overlay a slider on top of the widget. In the method we settled on, the user can grab a handle perpendicular to the surface and pull it in or out along the gradient of the scalar field. The manipulation is thus constrained to changing the value of the isosurface without changing the center of interest very much. This widget handle should work well in a traditional 2D environment as well, since it provides a simple way to map from mouse motion to manipulation of the surface value.

Since the widget can be obscured by the isosurface generated, we also considered interfaces that let one grab the surface directly. Our technique, when it detects a grab in space, attempts to find a nearby point on the surface and snaps the widget to that point. This seems to work well in practice and has been extended so that the user can grab in empty regions to see if they contain any non-contiguous isosurfaces. However, because grabbing into empty regions can be confusing, we did not incorporate this extension in the final version of the interface.

The finished version of the widget consists of a ball that allows free translation in space and generates the isosurface through its center, and a line representing the gradient vector at that point. Since a large handle on the gradient vector could obscure the data, the vector is instead highlighted only when it is grabbable, by displaying a ball at its tip.

THE CUTTING PLANE

The Cutting-Plane Algorithm

The other visualization technique we implemented was a direct-manipulation interface to cutting planes, shaded or contoured by the scalar fields through which they pass. The initial approach was to sample the field uniformly and shade the plane by these samples. However, this is inefficient on non-uniform grids, since data is lost in areas of high grid density unless the sampling rate is also very high.

However, one can imagine computing the intersections between the grid voxels and the cutting plane consisting of $Ax + By + Cz + D$, and generating polygons between these intersections. This will generate a cutting plane whose density varies according to the density of the grid in that region. The value at which the function of the plane, $f(x) = Ax + By + Cz + D$, is zero where the plane is located. Therefore, the plane can be considered an isosurface of that function. Because of this, we can use the same tracking algorithm we used in visualizing isosurfaces.

To use this cutting plane to visualize our original scalar field, we can shade it according to the scalar field, or we can generate contour lines on this slice.

The Cutting Plane Widget

A simple interface for direct manipulation of a bounded cutting plane is a rectangle on that plane. Our first attempt at creating interactive plane widgets was involved with an interactive toolkit for building 3D widgets [14], as shown in Figure 3. This widget contains considerable geometry, including a resize handle, a plane normal, an up vector, the projection of this up vector on the plane, and a representation of the space determined by the plane. This widget

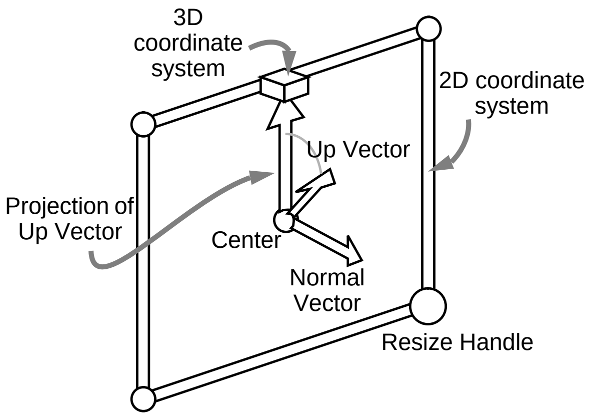

Figure 3: A widget which would hide scientific data

was too complex and obscured the data visualization, so we progressively simplified it to its present form, a rectangle composed of four thin lines.

The user can move the plane widget interactively by grabbing inside the widget and translating freely. All four sides of the rectangle highlight to indicate this selection. Note that the behavior of the algorithm to generate the cutting planes through the field is the same as in generating isosurfaces – it becomes ragged while it is being moved and fills out the entire plane when released.

The user can change the orientation of the plane by grabbing a single edge, which highlights to indicate that this is possible, and then translating it. The edge translates in the plane perpendicular to itself, maintaining contact with the hand and ensuring that the plane widget always remains rectangular. When the hand moves beyond a corner of the rectangle, the edges resize to keep the corner in contact with the hand.

We could also use the edges of the rectangle to clip the generated plane and view a selected region on it; we do not currently take advantage of this degree of freedom of the widget.

USER STUDY

Methodology

We ran a preliminary user study to examine the usefulness of these widgets. Because scientists who use visualization software generally perform high-level tasks such as exploration and interpretation, which are difficult to measure, we instead gave the subjects a short time to learn the system and then collected anecdotal reports.

The experimental group consisted of four scientists, two scientific illustrators and two visualization programmers.

They all work at NASA Ames and have considerable experience with scientific visualization.

The particular equipment configuration which we use consists of a Fakespace BOOM, which provides a high-resolution head-tracked stereo monochrome display, an SGI Skywriter for computation and rendering, and a VPL DataGlove with a Polhemus tracker for gestural interaction.

The visualization was of a scalar field, computed over a grid thirty voxels on each side and representing an inverse-square force (e.g., gravity) interaction between two bodies. The first task was to use the isosurface widget interactively to visualize this scalar field, and the second task was to do the same with the cutting-plane widget.

Two criteria were used to judge a technique's usefulness:

- Was the scientist able to understand the data?
- Would the scientist use the tool again?

The first criterion is crucial, of course, since understanding the data is the goal of scientific visualization. The second criterion is important because it tells us whether the scientist would like to use the tool. Through conversations with scientists and programmers, we determined that what appear to be beautifully-designed visualization tools are often left unused by scientists because they do not want to learn another tool.

Results

Hardware and Software Environment Problems Although the widgets were intended to be the focus of the experiment, the subjects reported being very frustrated by the hardware and by navigation through the virtual environment. Most of them complained about the monochrome display, the difficulty of using the BOOM to go where desired, and the jitter of the hand tracking. In addition, they had difficulty in perceiving stereo cues in the image and suffered from disorientation.

Some of these problems could have been solved by more powerful navigation techniques through the data, such as "flying" and "grabbing" the space [6]. The disorientation may have been due to the lack of perceptual cues and the difficulty of maneuvering through the environment.

The subjects also complained repeatedly that their "arms weren't long enough;" they would have preferred to be able to work with the widgets from a distance in order to gain a more global perspective on the effects of their manipulations.

This could be done through the inclusion of a “point and manipulate” interface, in addition to the direct grabbing one.

The Widgets Themselves When the subjects’ overall difficulties in using the VR environment were factored out, they had very positive reactions to the visualization widgets: although certain widget handles were initially difficult for the subjects to grab, after their first or second successful try, they experienced few difficulties.

Most of the subjects commented on how much faster the manipulation was than the tools which they normally used. Only one user commented on the degradation caused by the time-critical algorithm, so this algorithm seems to work well and unintrusively.

In the isosurface test, all subjects were able to understand the data being visualized. However, one subject complained that he could not understand the exact location of the bodies in the field, and another did not get a good understanding of the interactions between the two bodies.

Six of the subjects thought that the isosurface widget itself was good, though the other two wanted to reserve judgment until it was used on real data sets. There was one principal complaint: sometimes the numeric display of the isosurface threshold was hard to read or was obscured by the surface.

The subjects were more enthusiastic about the cutting-plane widget, although the range of responses was greater. A few subjects commented at length on the natural feel of the technique, but one had serious difficulties using the cutting plane. In general, users had difficulty selecting one edge of the widget rather than the entire plane. This could be easily solved by enlarging the pick region for the edges.

The edge-grabbing interface for specifying the rotation of cutting planes elicited the most positive response, with subjects calling it “cool” and “intuitive.” However, a few subjects requested numerical information on the position and orientation of the cutting plane, as well as techniques for orienting it along the principal axes. This could be done by adding handles, or perhaps by creating context-sensitive menus to change the parameters of the tools.

Two of the subjects, both scientists, commented that having these tools available would allow for new data visualization approaches not possible in their current workstation environment. They felt constrained by current interaction methods, and were interested in the potential power of this technology for data exploration.

CONCLUSIONS

Our guidelines for the design of 3D widgets for scientific visualization work well in practice: they produced good first-approximation solutions to the problem and helped speed the design process. However, user studies are still essential to fine-tune the behavior of the interface.

Time-critical algorithms are extremely important in virtual reality, and we are just beginning to understand how the algorithms used can influence interface design.

The hardware for VR still needs to mature, become less bulky and intrusive, and achieve greater resolution both of input and of output. As these problems are solved, many of the difficulties of using VR for scientific visualization should be alleviated.

FUTURE WORK

We are currently developing interfaces for other scientific visualization tools, and are examining the application of our scalar widgets to exploring unsteady scalar field visualization. In addition, we hope to explore other time-critical techniques including variable-resolution surfaces, incremental refinement, and viewpoint-dependent rendering.

ACKNOWLEDGEMENTS

Many thanks to Steve Bryson and Creon Levit for their help and suggestions, and to John West, Jeff Hultquist, Michael Gerald-Yamasaki, and Sandra Johan for their comments on the work in progress. Vee Hirsch, Ken Herndon, Scott Snibbe, Lars Bishop, Trina Avery, John Hughes, and Andy van Dam helped in the preparation and critique of this paper.

This work was supported by NASA Contact NAS 2-12961.

REFERENCES

1. G. V. Bancroft, F. J. Merritt, T. C. Plessel, P. G. Kelaita, R. K. McCabe, and A. Globus. FAST: A multi-processing environment for visualization of computational fluid dynamics. In A. Kaufman, editor, *Visualization*, pages 14–27, 1990.

2. J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–356, 1988.

3. J. Bloomenthal and B. Wyvill. Interactive techniques for implicit modeling. volume 24, pages 109–116, March 1990.

4. S. Bryson. Effects of lag and frame rate on various tracking tasks. Draft report.

5. S. Bryson and C. Levitt. The virtual windtunnel: An environment for the exploration of three-dimensional unsteady flows. In *Visualization '91*, pages 17–24, 1991.

6. J. Butterworth, A. Davidson, S. Hench, and T. M. Olano. 3DM: A three dimensional modeler using a head-mounted display. *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, 25(2):135–138, March 1992.

7. D. B. Conner, S. S. Snibbe, K. P. Herndon, D. C. Robbins, R. C. Zeleznik, and A. van Dam. Three-dimensional widgets. *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, 25(2):183–188, March 1992.

8. R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):65–74, August 1988.

9. W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In M. C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 163–169, July 1987.

10. S. S. Snibbe, K. P. Herndon, D. C. Robbins, D. B. Conner, and A. van Dam. Using deformations to explore 3d widget design. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):351–352, July 1992.

11. R. R. Springmeyer, M. M. Blattner, and N. L. Max. A characterization of the scientific data analysis process. In *Visualization*, pages 235–242, 1992.

12. R. M. Taylor, II, W. Robinett, V. L. Chi, F. P. Brooks, Jr., W. V. Wright, R. S. Williams, and E. J. Snyder. The Nanomanipulator: A virtual reality interface for a scanning tunnelling microscope. In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 127–134, Aug. 1993.

13. C. Upson, T. Faulhaber, Jr., D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. van Dam. The application visualization system: A computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 9(4):30–42, July 1989.

14. R. C. Zeleznik, K. P. Herndon, D. C. Robbins, N. Huang, T. Meyer, N. Parker, and J. F. Hughes. An interactive 3D toolkit for constructing 3D widgets. In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 81–84, Aug. 1993.